Security Considerations for Workflow Systems

S. Li, A. Kittel
GTE Information Technology
919 Hidden Ridge
Irving, Texas 75038
USA
{shunge.li,armin.kittel}@ap.bdi.gte.com

D. Jia, G. Zhuang
Intel Corporation
5000 W. Chandler Blvd.
Chandler, AZ 85223
USA
{debin.jia,guozhong.zhuang}@intel.com

Abstract

Security services for workflow systems are becoming increasingly important for cross-domain interoperability in insecure environments. Workflow interfaces and system components involve the whole spectrum of security services including authentication, authorization, access control, data confidentiality and integrity, audit, non repudiation, and administration. With the equipment of these services, workflow systems can be used in a broader range of enterprise applications. In this paper, we address many security issues in typical workflow systems and present a security model that utilizes the secure socket layer protocol and HTTPS tunneling mechanism. This model provides flexible security facilities that are suitable for application-to-application operations and data exchange over the Internet.

Keywords

Operations Support Systems, Workflow, Security Model, Security Management, Security Policy, HTTPS Tunneling, E-Business

1. Introduction

Security is a major challenge many operations support systems (OSS) are currently facing. One aspect of security concerns is how to protect OSS from being attacked by external sources while allowing ordinary business activities with authorized external customers or business organizations to be conducted. The so called e-business models, such as business-to-business e-commerce, business-to-consumer e-commerce, on-line reseller, and electronic supply chain, demand systems interoperability and data interchange across the boundaries of multiple security domains (also business domains or organization domains in most cases) separated by insecure environments. Company merger or acquisition also generates needs for cross-domain systems interoperability and/or integration. These demands apply as well to the area of workflow management [1], a key technology in OSS interconnection management.

Despite the existence of great demands for secure e-business and secure workflow management, so far security issues have not been well addressed, if not largely ignored, at OSS level. This is attributed to the low level of awareness as well as the lack of security infrastructure that is robust and yet flexible to use.

What secure e-business systems (including secure workflow systems) really need is an integrated security infrastructure that can offer such services as authorization, authentication, access control, data confidentiality, data integrity, auditing, non-repudiation, and security management and administration [3]. In order for this security infrastructure to be integrated well with an OSS infrastructure, these services must be supplied not only for system administration that traditionally provides system protection, but for system development as well. This requires that these services be available in the form of application programming interface (API) and included in the development cycle. Note that though important to a workflow system, resource management, which can effectively prevent denial of service attack, and exception handling, which can properly handle run-time system anomalies, should be ***built-in*** features of a workflow management system and not OSS-level services that can be offered to other OSS systems components.

A workflow system may engage multiple security domains. For example, in business-to-business e-commerce applications, data exchange takes place between two workflow systems that belong to different business/operational domains separated by firewalls. In Web-based e-commerce applications, an end customer may initiate a workflow process via a browser. In reseller model, a workflow engine may invoke a remote application at a reseller's site. Finally, a workflow system administrator (WFA) may want to remotely monitor workflow systems at production. In all these scenarios, some portions of the workflow systems may be exposed in insecure environments as disparate business domains are usually separated by public domain network such as the Internet.

The rest of the paper is organized as follows. Section 2 addresses many important security issues faced by modern workflow management systems and introduces some interesting concepts and mechanisms that may find their uses in security policy, secure transport protocol, access control, and transactional workflow. Section 3 surveys related work on workflow security model, security standards, and existing products. A security model particularly suitable for a variety of e-business models is proposed in Section 4. Section 5 concludes the paper with discussions and future work.

2. Workflow Security Issues

2.1 Workflow Systems Interfaces

In Workflow Reference Model (WRM) [2], standardized by Workflow Management Coalition (WfMC), five interfaces around workflow engines are defined. They are Process Definition Interface (Interface 1), Client Application Interface (Interface 2), Invoked Application Interface (Interface 3), Interoperability Interface (Interface 4), and Administration and Monitoring Interface (Interface 5) (Figure 1). These

interfaces provide a set of functions through which workflow systems direct workflow activities and control the interactions between workflow engines and other major components in the reference model.

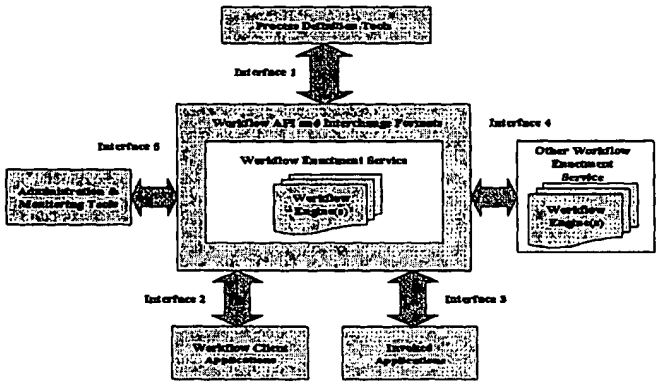

Figure 1: Workflow Reference Model

Workflow security can be addressed upon all the five interfaces, either in local domain or cross-domain. Consider the following nested workflow model (Figure 2). The second activity instance or task is actually another workflow process instance. The process definition of *Workflow A* logically contains the process definition of *Workflow B*. Multiple scenarios involve security. First, both workflow definitions may be physically stored in disparate locations, and their run-time process instances are not necessarily executing in the same domain. Second, the parent workflow and the child workflow may be handled by two workflow engines in different workflow systems, respectively. Third, both definitions may have different ownership, leading to the requirement for access control mechanism. A real world example of this scenario would be a service provisioning workflow where a customer's credit is checked before an order can be further provisioned. The credit validation workflow can take place at a credit bureau site located in a separate security domain (Figure 2).

Similar scenarios are true for other interfaces as well. A Web client outside a firewall may want to initiate a workflow service (Interface 2). A workflow system may provide a remote service located in another security domain (Interface 3). Workflow engines, servers that provide enactment services, may be either in the same process address space, or in different processes of the same local platform, or in disparate remote processes. Interface 4 provides functions of the interoperability between workflow engines that may reside in two security domains. Interface 5 provides administrative operations that may be performed in a networked environment such as the Internet.

Security involving cross-domain interfaces should generally cover the following aspects: access control, level of security services, security policy, and security implications on system performance and anomaly management.

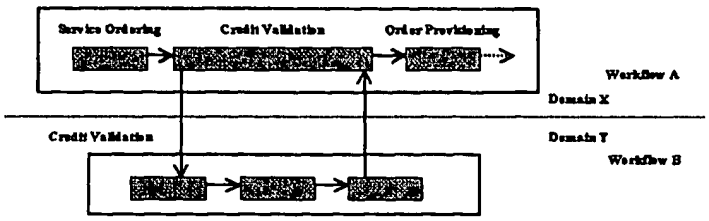

Figure 2: A Cross-Domain Nested Workflow

2.2 Access Control

The security issues with workflow systems are not just limited to cross-domain interfacing. Access control mechanisms are needed at both workflow level and task (workflow activity) level, even within a single security domain.

First, there is a concept of ownership for each process definition and process instance as well as for each process activity and process activity instance. This particularly makes sense for hierarchical workflow, considering the fact that a sub-workflow may have a set of ownership that is totally different from its parent workflow. Ownership determines the obligation of process creation and overall responsibility of any consequences resulting from all operations on a process. Therefore, the owner of a process or its instances must be careful in granting access permissions to proper parties. Rules that define how to transfer or delegate the access permissions between parent workflow and child workflow and mechanisms that enforce these rules are desired in workflow systems.

Second, data in a workflow system fall into several categories, including application-specific data, logs, workflow relevant data (e.g., for state transition and internal queue management), workflow control data (e.g., for session management and interoperability protocol), audit trails, process definitions, process instances, and performance data (for monitoring and administration purposes). Also in a workflow system are multiple identities: process creators, WFAs, workflow engines, workflow participants (initiation clients and applications), and process definition repositories. In a cross-domain setting, the question "can a client perform an operation on a workflow application?" would generally lead to the definition of an access matrix for every data category and every identity in a workflow system.

Many access control scenarios can be covered in such a matrix. For example, a WFA has the privilege to specify who has appropriate permissions to read, write, and/or execute a workflow and its associated applications. Similarly, audit trails are generally available only to WFAs, not to ordinary workflow participants. In some applications, business-sensitive data (e.g., social security number, credit card number, or account balance) should remain read-only and encrypted, even to a WFA. Even if a WFA can view application-specific data, he/she cannot overwrite them. Likewise, a Web client who is outside the security domain can only read and execute a workflow, but cannot change it.

Access control is generally attained by means of authentication and authorization. Workflow systems need authentication mechanism to assure that only authorized users or software modules perform appropriate operations on workflow and tasks. Since a workflow object is often an "intermediate" object delegating a client's request to a server and a server's response back to a client, the workflow object should be able to support at least "combined privilege delegation" and in higher accountability situation to support "traced delegation" [5]. The privilege heritage mechanism used in CORBA object invocation [6] can be utilized to accomplish access control provisions.

2.3 Security Level and Policy-based Security Management

Ensuring workflow security does not come free. It has a big performance impact on workflow systems. As a matter of fact, data confidentiality and data integrity depends largely on cryptographic algorithms (e.g., public-key encryption algorithms), which are usually computation-intensive tasks. Therefore, workflow systems need to seek a balance between the level of security and the overall system performance in terms of time and space efficiency.

Multiple levels of security services for workflow systems can be defined under various networking circumstances. For example, an intranet is usually within a protected domain and therefore may only need authorization and authentication services whereas in the Internet environment, data confidentiality and integrity services must be provided. Below are a number of security services that could be applied to workflow systems in increasing levels of security and finer granularity of security control:

- No security services,
- Firewall protection only (system-level security),
- Authorization only (login and password),
- Simple authentication (digital certificate) at session establishment,
- Authentication at each round of message exchange,
- Authentication plus optional encryption of individual messages, and
- Authentication plus encryption of all messages.

Offering multiple levels of security for workflow systems allows appropriate relaxation of security requirements for performance. Security policies can be introduced to balance between the levels of security and system performance. A time-based policy specifies when or how frequently a security check (e.g., login process) needs to be enforced and a security service needs to be performed. Examples of time-based policy include automatic logout of a process that has been idle for certain period of time, change of a password upon its expiration, and renewal of a *service contract*. In workflow systems, systems failures or security violations could cause no response from applications. Timeout mechanism is often used against denial of service attack to assure system robustness. Similarly, attacks that feature exhaustive consumption of workflow systems resources (such as memory and file

descriptor) and significant slow-down of system performance must be prevented effectively.

Admission control, a mechanism to protect a system from serving more clients or applications than it could handle, can provide additional help. Knowing the perceived resource usage or requirement of an entity (client or application), admission control can determine, based on current resource utilization, whether to accept the incoming entity to the system. Sometimes, preemption is necessary to accommodate a new entity by kicking some existing entity out of the system. However, guidelines need to be followed to guarantee fairness, or to reflect some preset policies. Such policies could be FIFO, LIFO, or priority-based.

But that's not sufficient! Just because a Web client is authorized to initiate a workflow doesn't mean the client can do so unlimited. Otherwise, the workflow engine would run out of resources rather quickly and cause degradation of its services critical to other useful activities. Therefore, in addition to a security profile [3] that specifies access control permission and authorized clients, a *service-level contract* that quantifies the amount of operations (e.g., workflow initiation) allowed to be performed per time unit is also indispensable.

We can define following security policies.

- Per-session authentication means that involving parties' certificates are verified at the time of communication session establishment, whereas per-message authentication means that for every message exchanged between two parties, the sender's certificate is checked.
- On-demand data encryption means a message is encrypted only when needed, based on performance requirements or other indications, whereas per-message data encryption ensures data confidentiality and data integrity for all messages exchanged.
- Workflow-level access control involves grant and revoke of read/write/execute permission and transfer of ownership privilege at workflow level whereas task-level access control involves same operations at a finer level of granularity. Note that security checking can be implemented as a precondition, either at workflow level or at task level.
- Frequency-based security enforcement policy specifies a frequency f at which security check (e.g., authentication) is enforced. Frequency f indicates that security enforcement takes place every f message exchanges. It is a compromise between two extreme cases: security enforcement only once ($f = \infty$) and security enforcement every time ($f = 1$).
- Time-based security enforcement policy specifies a time interval t. Security check is enforced every so often. Interval t should be dynamically configurable.
- Admission control policy can be defined based on the number of clients allowed to enter a workflow system or on the workflow system load.
- Contract-based security policy requires a contract to be signed by two communicating parties. A contract is an agreement between both parties that specifies the maximal throughput either party can afford and/or the minimal

throughput either party needs to provide. It can also include other quantitative parameters such as Quality of Service (QoS) parameters and other qualitative policies that have performance impacts.

The concept of service-level contract is important as well as interesting in that it contains quantitative parameters for negotiation during secure channel establishment in a secure transport protocol. Contracts can be negotiated dynamically or adaptively at run time, based on network states or some performance indications. This concept is generic and can be applied to non-workflow systems as well.

2.4 Transactional Workflow Security

Transactional workflow requires that workflow systems maintain ACID properties (atomicity, consistency, isolation, and durability). A transaction is either performed in its entirety or not performed at all. The states and transitions of a workflow system should remain consistent before and after a failure occurs. Since rollback operations degrade overall performance, a workflow system needs to keep the number of failures at a minimum. Let's consider a failure scenario due to security violation that prevents a workflow engine from proceeding workflow activities. Because transactional workflow is costly to rollback, rather than undoing what have been done successfully, we authorize some proxies to continue a workflow process on behalf of a principal. A principal is a human user or system entity that is registered in and authentic to the system.

Of course, the workflow system must maintain audit trails that have all the records of operations a proxy has performed on behalf of a principal in order to trace any security violation in case a problem occurs. An active entity, either a principal or a proxy acting on behalf of a principal, must establish its right to access objects in the system.

Consider a classic example of workflow application where a health insurance claim is submitted for processing. This transactional workflow consists of three sequential tasks: claim report, claim verification, and claim approval. After the verification of patient information, the claim form is sent to a manager for approval. Suppose the manager is on vacation. The claim processing would halt until the workflow is timed out, in which case the transactional workflow would be aborted, causing two previous tasks (claim report and claim verification) to be undone. If an assistant of the manager was assigned to sign the claim on behalf of the manager (the principal), the transactional workflow could be successfully completed.

3. Related Work

3.1 WfMC Simple Workflow Security Model

Workflow security is a young research area that is still in its infancy. In 1998, the *Workflow Management* Coalition issued a white paper on workflow security [3], the first document issued by WfMC on this topic, which proposes a simple workflow security model that focuses on cross-domain interoperability security. Defined in this

technical document are such security operations as authentication at the level of peer workflow services, data confidentiality and data integrity on the data content of the interoperability protocol, and audit to provide consistent audit data. An important feature of the model is the introduction of the concept of *common security profile*, which consists of three parts: (1) security measurement information (identity, encrypting algorithms and keys); (2) information about how security measures should be used; and (3) information about operation authority. A common security profile is mutually agreed and consistently maintained by peer workflow enactment services and is used to control the way in which security is applied during interoperation.

There are two basic ways to build and maintain security profiles. One is to establish a profile using a manual procedure; the other is to provide automatic support through negotiation between the participating parties. In previous section, we introduced the concept of service-level contract as a mechanism for enforcing security policies and negotiation during secure transport channel establishment. Here, we further propose to extend common security profiles to include service-level contracts that quantify the amount of allowable operations and QoS parameters. Both common security profiles and service-level contracts would be a part of workflow control data [2], which are subject to security check through access control.

3.2 Web-based Security Model for OSS

Many enterprise-wide operations support systems have been migrating to become Web-enabled due to the explosion of the World-Wide-Web in recent years. Figure 3 depicts a commonly used security model for OSS.

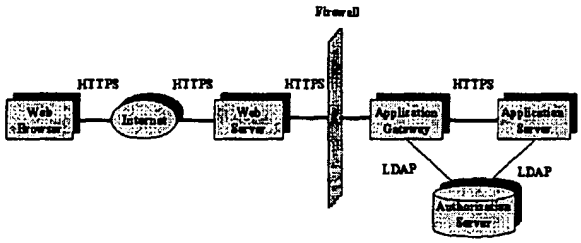

Figure 3: Web-based Security Model for OSS

In this security model, a Web server is placed outside an enterprise firewall to receive and decrypt HTTPS requests from a client that uses a browser. The server first authenticates the client's identity. Once the authentication succeeds, it forwards the HTTPS requests and the client's identity information to the Application Gateway behind the firewall. The Application Gateway does further authentication on the client's identity against a list of authenticated users stored in the Authorization Server database using Lightweight Directory Access Protocol (LDAP) [7]. If the authentication process is successful, the Application Server returns to the client a Web page with links to accessible applications.

While this Web-based security model can satisfy many enterprise systems' needs, the client programs are restricted to be applets. The model is not applicable to some e-business models we described earlier, such as the on-line reseller model, where application-to-application rather than applet-to-application secure transport service is needed.

3.3 Current Status of COTS Products on the Market

COTS (Commercial Off-The-Shelf) products offer standard-compliant, quality implementation of security services for use in distributed computing environments. Integrating these products into existing workflow systems enables easy and quick provisioning of security services. Current COTS workflow products can roughly be categorized as CORBA-based workflow systems and non CORBA-based workflow systems.

CORBA [11], as an enabling distributed computing technology, has had worldwide acceptance. There have been research efforts leading to CORBA-based workflow systems. The OMG document entitled "jFlow – Workflow Management Facility" [5] proposes to WfMC IDL specification for workflow management. The specification is based on WfMC released OMG-IDL binding for its Client Application Programming Interface and Interoperability Interface [5]. Its security support relies on "CORBA Security Service" [6]. While CORBA Security Service addresses a number of security issues and security functionality for general CORBA-based systems, IIOP itself does not provide data encryption unless used with some security packages that provide it, such as Secure Socket Layer (SSL) [10], one of the most popular security standards. Unfortunately, IIOP over SSL traffic is still not a standard traffic that most firewalls accept.

CORBA-based COTS products from some leading industry vendors including Iona and Inprise offer the following features:

1) IIOP over SSL solution (for both applet and application),
2) IIOP over HTTP (for both applet and application), and
3) IIOP over HTTPS (applet only)

The first solution requires a dedicated IIOP port to be open at all firewalls involved in the communication path. This requirement puts an additional configuration effort at senders' sites and also creates a security hole (though under strict control) at the senders' firewalls. Although CORBA 3.0 [8] specifies dedicated IIOP ports for future firewalls, it is up to vendors' effort to conform to the standard, and until then most firewalls do not support default IIOP traffic automatically. The second solution does not provide encryption. The third solution only applies to Web-based applications (applets).

SSL is a natural choice for non CORBA-based workflow systems due to its simplicity and yet complete offering of authentication, confidentiality, and integrity services. Originally introduced to provide security for Web browsers by encrypting HTTP connections, SSL now serves as a vehicle for providing security for general

Internet services [9]. SSL hides the details of encryption and authentication in the socket library calls, making the programming on the Internet much easier.

Vendors such as Phaos Technology and IBM provide SSL-level APIs. However, SSL traffic alone would be blocked by firewalls, unless it is HTTPS traffic. HTTP and HTTPS are among the standard traffics that most firewalls allow to get through. HTTPS tunneling is an ideal mechanism for penetrating firewalls while maintaining data confidentiality and data integrity. Therefore, what is really needed for a security infrastructure is an API for HTTPS tunneling. Note that Java Development Kit (JDK) only offers an API for HTTP, not for HTTPS.

3.4 Simple Workflow Access Protocol (SWAP)

Another standardization effort in workflow management is Simple Workflow Access Protocol (SWAP) [4]. Some of the notable features are the use of XML [13] as a uniform format for data exchange. This protocol addresses Interface 4 interoperability, but fails to address any security issue. Since it relies on HTTP as a transport messaging service, it is easy to make security extension by using HTTPS instead of HTTP. Next section presents a secure transport service, namely an API for HTTPS tunneling service developed at GTE.

4. A Security Model for Workflow Systems

The security model proposed in this paper is centered on the SSL protocol and the HTTPS tunneling mechanism. Our goal is to provide a secure transport service for e-business across firewall-protected business domains. Taking advantage of the fact that most real-world firewalls are open to HTTPS traffic without any additional firewall configurations, HTTPS tunneling is an ideal security mechanism to meet our goal. However, there is currently lack of APIs for HTTPS connection management and messaging in JDK as well as in COTS products. Our security infrastructure, namely HTTPS tunneling API was hence developed to fill up this gap.

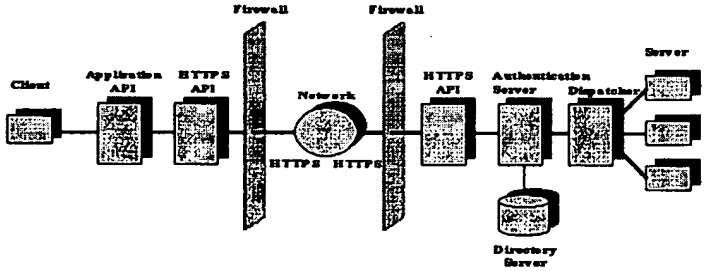

Figure 4: Proposed Security Model for Workflow Systems

HTTPS, or HTTP over SSL, is achieved by putting HTTP packets into secure sockets created using secure socket library. When using HTTPS tunneling, an

application encapsulates its data into an HTTP packet before sending it over a secure socket channel, and extracts the data from the HTTP packet after receiving it from the other end of the channel. The advantage of using HTTPS tunneling lies in the fact that most firewalls are HTTPS-ready and allow HTTPS packets to pass through without any additional configuration.

Figure 4 illustrates our proposed security model for e-business applications based on HTTPS tunneling mechanism. There are a client-side firewall and a server-side firewall, separated by an insecure network such as the Internet. Application-specific data are HTTPS-encapsulated and tunneled through both firewalls. Secure communication details are hidden from ultimate clients and servers. At the server side, there is an authentication server that is responsible for extracting senders' identity information (such as the distinguished name) from their certificates, checking their validity, and matching them against a list of authorized clients stored in a directory server (e.g., LDAP server). Upon successful authentication, data is dispatched to an appropriate server for processing.

This model is more than just HTTPS tunneling and certificate authentication; it can provide higher-level programming abstraction by supporting application-level API. Here is a typical scenario where a client wants to request a service offered by a server located in a separate security domain. Suppose both a client and a server agree upon an interface for a particular service. The server is responsible for the implementation of the interface and the client for supplying input data and receiving output results. A proxy that handles data marshaling and HTTPS tunneling tasks can be constructed at the client side. The client does not know, and does not need to know, whether or not the server is in the same security domain as itself. All it needs to know is how to communicate with the proxy according to the interface specification. Hence, the proxy serves as an abstraction of the lower-layer transport and remote service implementation. There is one proxy per service implementation. Since the tasks performed by a proxy are fairly standard, compiling an interface automatically generates the proxy code, the "stub" to the client.

In practice, interfaces are application-specific APIs; they are defined by application programmers and built on top of the HTTPS tunneling API to provide higher-level business-oriented functions such as customer login, session information management, customer information management, etc.

The model can also support CORBA-based applications for electronic data interchange, by making application-specific APIs CORBA-aware. Although this model is intended for application-to-application secure data exchange, it can be used in Web-based secure data exchange (applet-to-application) as well. The proposed security model has the following features:

1) It provides a standard-based (compliant to SSL 3.0) and simple HTTPS tunneling API that is easy to use, platform independent due to pure Java implementation, and reliable. No vendor product on the market offers this HTTPS tunneling at API level.
2) Firewalls are easy to configure. The only requirement for firewall setup is to open for HTTPS messages, which is a widely accepted firewall policy.

3) Its architecture is flexible in that the model supports both Web-based and non Web-based applications, as well as CORBA-based applications. Its security infrastructure is generic and can easily become a reusable component.
4) The architecture supports multi-threading and allows concurrent use of the API.
5) It provides such security services as authentication (client side and server side), confidentiality, and integrity.
6) Application-level APIs can be "compiled" automatically to produce proxy codes.

5. Conclusions

The HTTPS tunneling API has been implemented and successfully used in some GTE's projects that offer Unified Messaging (UM) and Internet Call Waiting (ICW) services being deployed nation wide. Preliminary experiments show that performance is satisfactory for business requirements. For example, in the existence of a firewall, end-to-end (from client to server) function calls take about 2.4 seconds, including the times spent in SSL connection establishment, data encryption and decryption, marshaling and unmarshaling, HTTPS tunneling, and authentication via LDAP. Compared with the 5-minute-per-transaction business requirement, the time spent in HTTPS tunneling is negligible. This further demonstrates the feasibility and simplicity of the security model we proposed.

In future, we plan on making the security infrastructure a reusable component that can meet most enterprise-level needs for secure data exchange. Easy configurability and extendibility of application-specific APIs are what we want to achieve in componentization. We also plan on providing XML support at API level to maximize the interoperability. Although multi-threading is currently supported, improvement can be made to ensure that it is scalable as the number of incoming HTTPS requests increases drastically.

There is currently no vendor workflow product on the market that offers full-fledged security services. Incorporating the security infrastructure into our workflow system framework would establish a test bed that would facilitate the future research activities on workflow security.

Security services need to be leveraged both at workflow system level and at the transport protocol level. Although providing authentication, confidentiality, and integrity, current secure transport protocols have not taken into account the performance impact of security services on communication systems. Studying the relationship between various security policies and systems performance would benefit the establishment of guidelines on selecting appropriate security levels under certain circumstances. The concept of service-level contract introduced in Section 3 may be helpful for this purpose. For certain applications, security requirements may be relaxed in exchange for better performance. Therefore, the quantitative tradeoff between levels of security services and QoS needs to be studied via experimentation. Service-level contract provides a simple and powerful mechanism for security service specification, and for QoS negotiation and run-time adaptive QoS control in secure transport protocol design. It is our plan to design a secure transport protocol

that addresses the relationship among workflow system performance, security policy, and QoS, and to conduct experiments that give answers to these questions.

References

[1]. Peter Lawrence (Editor), "Workflow Handbook 1997", Johnson Wiley & Sons Ltd.

[2]. David Hollingsworth, "Workflow Reference Model", Issue 1.1, WfMC-TC-1003, November 1994.

[3]. Workflow Management Coalition, "Workflow Security Considerations – White Paper", Issue 1.0, WfMC-TC-1019, February 1998.

[4]. K. Swenson, "Simple Workflow Access Protocol (SWAP)", Internet Draft, August 1998.

[5]. Shunsuke Akifuji, et al., "Workflow Management Facility (jFlow)", Revised Submission, OMG Document No. bom/98-03-04, 1998.

[6]. Object Management Group, "CORBAService: Common Object Service Specifications", 1997.

[7]. W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol", RFC 1777, March 1995.

[8]. Jon Siegel, "Component and Object Technology: A Preview of CORBA 3", IEEE Computer, Vol. 32, No. 5, May 1999.

[9]. Derek Atkins, et al, "Internet Security: Professional Reference", 2nd Edition, New Riders Publishing, 1997.

[10]. A. Freier, P. Karlton, and P. Kocher, "The SSL Protocol Version 3.0", draft-ietf-tls-ssl-version3-00.txt, November 1996.

[11]. Object Management Group, "The Common Object Request Broker: Architecture and Specification", Revision 2.2, February 1998.

[12]. Rolf Oppliger, "Internet Security: Firewalls and Beyond", The Communications of the ACM, Vol. 40, No. 5, May 1997.

[13]. World Wide Web Consortium, "Extensible Markup Language (XML)", http://www.w3c.org/XML/.

[14]. R. Fielding, et al, "Hypertext Transport Procotol – HTTP 1.1", RFC 2068, January 1997.

[15]. Thomas Bauer, Peter Dadam, "Efficient Distributed Control of Enterprise-Wide and Cross-Enterprise Workflows", Proceedings of Workshop Informatik '99: Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, Paderborn, Germany, October 6, 1999.

Biography

Shunge Li is a technical lead and principal member of technical staff of the Architecture and Design for Advanced Internet Services Department. His team is responsible for building reusable components and infrastructure for enterprise systems. Mr. Li received his Ph.D. degree in computer science from Purdue University in 1997. His research interests include distributed multimedia systems and

applications, networking and communications, workflow management, and electronic commerce. He is a member of IEEE, IEEE Computer Society, and Upsilon-Pi-Epsilon honor society.

Debin Lia is a senior software engineer in the Home Product Division at Intel Coporation. He worked at GTE as a senior systems engineer before joining Intel Corporation. Mr. Jia holds a master's degree in computer science.

Guozhong Zhuang was born in 1966. He received a B.S. in Mathematics from Nanjing University in 1988, a M.S. in Computational Mathematics from Chinese Academy of Sciences in 1991, and a M.S. and a Ph.D. in Computer Science from Purdue University in 1996 and 1999, respectively. His major research areas include computer graphics and visualization, numerical analysis, multivariate control systems and approximation theory. He is a software engineer at Intel Corporation.

Armin Kittel is Director of Architecture and Design for Advanced Internet Services as part of the Information Technology organization at GTE. His responsibilities include operations support systems (OSS) infrastructure for the Global Network Infrastructure, GTE's broadband backbone network and for advanced IP-based services launch. These efforts supported GTE's massive build-out of its broadband network and the launch of a new line of business in GTE Internetworking. Before being appointed to his current position he worked on OSS infrastructure for the initial Local Number Portability rollout. Before joining the Information Technology organization he worked at GTE Laboratories as a member of the Software Systems Laboratory designing and implementing an integrated network management system for GTE's entire network. Mr. Kittel received his Ph.D. in Electrical Engineering from the Georgia Institute of Technology in 1992 and his Master's in Computer Engineering from the Technical University at Braunschweig, Germany in 1987. He is a member of the IEEE Computer Society.